JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 11 (2019) 91-95
Wysłane: 2018-12-07
Przyjęte: 2018-12-16

Analiza zastosowania narzędzia Cucumber w testowaniu aplikacji

Illia Herman*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Niniejsza artykuł przedstawia analizę narzędzia do tworzenia testów automatycznych o nazwie Cucumber oraz opisuje jak dane narzędzie jest wykorzystywane w praktyce. W ramach artykułu stworzono kilka przypadków testowych, aby zweryfikować działanie obranego podejścia do testowania w programowaniu w językach Java i Kotlin, atakże został porównany czas wykonania testów w obu językachz rodziny języków obiektowych.

Słowa kluczowe: Cucumber; Gherkin;BDD; Java; Kotlin.

*Autor do korespondencji.

Adres e-mail: illiaherman8@gmail.com

Study on applying the Cucumber tool in testing applications

Illia Herman*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The paper presents an analysis of the Cucumber automated test tool and describes how this tool is used in practice.As part of the paper, several test cases were created to fully verify the operation of the chosen approach to testing in programming in Java and Kotlin languages also the time of testing were compared in both languages from the object-oriented language family.

Keywords: Cucumber; Gherkin; BDD; Java; Kotlin.

*Corresponding author.

E-mail address: illiaherman8@gmail.com

1. Wstęp

Proces testowania jest niezbędną częścią procesu wytwarzania oprogramowania,które jest wykorzystywane dla potrzeb użytkownika końcowego.Celem testowania oprogramowania jest uniknięcieróżnego typu usterek wytwarzanego produktu i organizacji maksymalnie komfortowej logiki jegoużycia. Testowanie pomaga zweryfikować poprawność działania oprogramowania i zebrać informacje na jego temat,aby móc obserwowaćwystąpienie wad i problemów. Dlatego zanim gotowy produkt trafi do użytkownika, należy pamiętać o testowaniu jako koniecznym warunku wytwarzania jakościowego oprogramowania.

Z punktu widzenia technicznego testowanie dzieli się na manualne i automatyczne[9]. Testy automatyczne, w przeciwieństwie do testów manualnych, bardziej opierają się na programowaniu i takie podejście wymaga mniejszych nakładów czasu a przede wszystkim pieniędzy [11]. Obecnie istnieje mnóstwo narzędzi do prowadzenia testów automatycznych, zagadnienie testowania aplikacji jest jednym z podstawowych zagadnień tworzenia różnego typu oprogramowania.

Tematem podjętym w niniejszym artykule jest analiza jednego z najpopularniejszych rozwiązań do tworzenia testów automatycznych o nazwie Cucumber. Narzędzie to posiada dużąbazę użytkowników, dokumentacji technicznej, dość częste aktualizacje, jest ciągle rozwijane i wspierana jest coraz większa liczba języków programowania. Cucumber opracowuje język naturalny i zrozumiały dla biznesu, co rozwiązuje typowy problem podczas komunikacji pomiędzy programistą, czyli twórcą aplikacji i klientem biznesowym. Wynikiem takiego podejścia jest łatwa dokumentacja testów, zwiększona ogólna wydajność i co najważniejsze szybsze dostarczenie produktu w postaci aplikacji do klienta.

2. Cel badań

Celem niniejszego artykułujest oszacowanie czasu wykonania testów w oparciu o narzędzie Cucumber w takich językach programowania jak Java i Kotlin.

Hipoteza badawcza postawiona w pracy brzmi: „Nie ma istotnej różnicy w czasie wykonania testów w narzędziu Cucumberdla języków programowaniaJava i Kotlin”.

3. Przegląd literatury

Temat automatyzacji testówstaje się coraz popularniejszy wśród twórców oprogramowania komputerowego, a tym bardziej dyskusja na ten temat toczy się w zespołach korzystających z metodyk Agile [10] bądź DevOps. Jeżeli chodzi o taki rodzaj testów nie można nie zwrócić uwagi na tak zwane testy behawioralne które są częścią BDD (z ang. Behavior - driven development) [8]. Składnia testów behawioralnych oparta jest o język półformalny, który pozwala pisać czytelne i zrozumiałe testy nawet dla osób niezwiązanych z branżą IT.

Zgodnie z definicją „ *Behaviour - Driven Development polega na wdrożeniu aplikacji poprzez opisanie jej zachowania z perspektywy interesariuszy” [1].*

Grupa badawcza z uniwersytetu RWTH Aachen w swojej pracy [2], testując różnego typu narzędzia BDD wspomina o narzędziu Cucumber, jako jednym z najpopularniejszych i powszechnie stosowanych frameworków tego typu oraz wyróżniają możliwość pisania tak zwanychhistoryjek

użytkownika i scenariuszy dla przypadków testowych, które można wykorzystać jako podstawę do tworzenia automatycznych testów akceptacyjnych, do których stosuje się badanenarzędzie.

Analizując literaturę na temat Cucumber warto również odnieśćsię do pracy Olav Undheim [3], która poświęcona jest testowaniu, a szczególnie do badania przeprowadzonego przez Olav Undheim, w którym wzięły udział trzy grupy składające się z dwunastu osób każda. Przed rozpoczęciem badania uczestnicy mieli odpowiedzieć na dwa pytania:

1) jakie mają doświadczenie w programowaniu;
2) jakie mają doświadczenie z Cucumber.

Otrzymane wyniki autor przedstawia w formie wykresów.
Z wykresów wynika, że:

1) żaden uczestnik nie miał więcej niż 2 miesięcy doświadczenia komercyjnego w programowaniu. Niewielka liczba osób ma krótkie komercyjne doświadczenie w IT, większość zetknęła się z programowaniem na studiach lub w czasie wolnym;
2) tylko kilka osób słyszało o Cucumber jeszcze przed eksperymentem, ale nikt z niego wcześniej nie korzystał.

Celem badania było opracowanie przez grupę zestawu nieskomplikowanych przypadków testowych oraz odpowiedzenie na parę pytań dotyczących pracy z Cucumber.

Pierwsze pytanie dotyczyło składni Cucumber i grupa C miała najbardziej zróżnicowane odpowiedzi, ale większość nie wykazała większych problemów w zrozumieniu składni narzędzia. Jedynie dwie osoby z grupy C uznało zadanie za raczej trudne.

Kolejne pytanie dotyczyłopotrzeby dodatkowych wskazówek podczas pisania testów. Tutaj głosy są bardziej zróżnicowane. W grupie A widoczna jest niewielka przewaga osób, chcących uzyskać dalszą pomoc (2 osoby więcej), w grupach B i C większość nie potrzebowałaby dalszego wsparcia.

Ostatnią kwestią, poruszaną przez badanie, była chęć pisania testów w wolnym formacie, to znaczy bez użycia pewnej składni w postaci słów kluczowych. W tym przypadku zdecydowana większość nie wyraziła takiej chęci. Kilkoro uczestników każdej z grup nie wyraziło zdania.

4. Narzędzie Cucumber

Jest to narzędzie na licencji *open source*[4] służące do wykonywania automatycznych testów funkcjonalnych.Zostało opracowane oraz stworzone przez Aslak'a Hellesøy'ac [5],[6] i jest cały czas rozwijane. Stosuje się je głównie jako wsparcie dla BDD [1], gdzie cała aplikacja budowana jest z komponentów, tak zwanych historyjek użytkownika. Do napisania testów automatycznych w Cucumber wykorzystuje się specjalny język Gherkin [7]. Obecnie Gherkin obsługuje ponad 60 języków naturalnych, w tym język polski. Gherkin rozwiązuje od razu dwa zadania: tworzenia dokumentacji technicznej projektu i oczywiście tworzenia testów zautomatyzowanych. Jak każdy język, posiada swoją składnię oraz strukturę, a opisuje się za pomocą już wspomnianych pewnych słów kluczowych. Mimo tego, że opcja pisania historyjek użytkownika w języku polskim jest dobrze opracowana oraz może się okazać bardzo przydatną i komfortową w oddzielnych przypadkach, to ze względu na uniwersalność języka angielskiego oraz jego wykorzystanie w dokumentacjach technicznych, zaleca się pisać testy szczególnie w języku angielskim.

Słowa kluczowe [7] które wykorzystują się do opisania historyjek użytkownika przedstawione są w tabeli 1.

Tabela 1. Słowa kluczowe w języku Gherkin.

Słowo kluczowe	**Do czego służy**
FEATURE	Słowo kluczowe które zawsze rozpoczyna plik testowy (wyjątki które mogą poprzedzać Feature tokomentarze oraz tagi). Opisuje nazwę przypadku testowego bądź funkcjonalności
AS, I WANT, IN ORDER	Formalnie nie są słowami kluczowymi, stosują się do opisu dowolnie formowanym. As — definiuje rolę, I Want — pożądana funkcjonalność, In order — rozszerza kontekst czynności w I Want.
SCENARIO	Scenariusz przypadku testowego. Kroki które należy wykonać, aby przetestować pewną funkcjonalność
GIVEN	Założenia początkowe, które przygotowuje scenariusz do wykonania
WHEN	Warunek, który wykonuje pewną akcję, na przykład na GUI (graficzny interfejs użytkownika)
THEN	Wynik scenariusza. Opisuje kroki sprawdzające jego prawidłową realizacją
AND	Dodatkowe założenia bądź dodatkowe warunki. Najczęściej używa się do rozszerzenia pewnego słowa kluczowego (Given, When, Then) o kolejne kroki
BUT	Rzadko stosuje się w tworzeniu historyjek, służy głównie do dodawania negatywnych komentarzy albo opisuje warunek, który jest przeciwny do założenia
BACKGROUND	Pewny warunek, który należy spełnić przed wykonaniem scenariusza
SCENARIO OUTLINE	Nazwa skomplikowanego scenariusza. Taki scenariusz w zależności od ilości danych wykonuje się wielokrotnie
EXAMPLES	Zestaw pewnych danych, które po kolei będą używane do następnych iteracji scenariusza

5. Cucumber w praktyce

Niniejszy rozdział opisuje metodę badawczai proces analizy narzędzia Cucumber pod kątem wykonania testów w językach Java i Kotlin.

5.1. Plan badań

Plan badań oparty jest na sprawdzeniu funkcjonalności oraz konfiguracji Cucumber w językach programowych Java i Kotlin, a także porównaniu czasu trwania wykonania testów, który jest liczony w sekundach, w obu językach z rodziny języków obiektowych.Realizacja badań została przeprowadzonana dwóch urządzeniach o różnych parametrach w celu otrzymania wiarygodnych wyników.

Pierwszy etap badania polegał na konfiguracji środowiska wykonawczego oraz podłączeniu niezbędnych do wykonania testów bibliotek.

Kolejnym etapem było stworzenietrzech aplikacji testowych z szeregiem różnychfunkcjonalności oraz przygotowaniu zestawu testów w Cucumber. Następnie takie testy zostały przeprowadzone, a wyniki w postaci czasu wykonania testów zapisane.

Trzecim a jednocześnie ostatnim etapem byłoporównanie oraz omówienie otrzymanych wyników.

5.2. Narzędzia i technologie wykorzystane podczas implementacji badania

Java w wersji 10 – o tym języku programowania chyba wiedzą nawet ludzie nie związani z IT. Dany język zorientowany jest obiektowo oraz jest częścią tak zwanych języków programowania wysokiego poziomu. Obecnym właścicielem Java jest Oracle Corporation. Język działa na wirtualnej maszynie JVM (z ang. Java Virtual Machine)

JVM w wersji 1.8 – maszyna wirtualna Java to środowisko które wykonuje kod bajtowy Javy.

Kotlin w wersji 1.2 – język programowania który powstał w porównaniu do innych języków dość niedawno, w 2011 roku. Również działa na wirtualnej maszynie Java.

IntelliJ IDEA wersji 2018.2.5 x64 Ultimate – środowisko programistyczne dla języków opartych na JVM, właścicielem, którego jest znany producent oprogramowania JetBrains.

Selenium WebDriver w wersji 3.14.0 – połączenie serwera Selenium z sterownikiem zapewniającym komunikacje z konkretną przeglądarką.

Apache Maven wersji 3.6.0 – narzędzie które automatyzuje budowę projektu.

JUnit w wersji 4.2 - jeden z najczęściej wykorzystywanych frameworków do przeprowadzenia testów jednostkowych w języku Java.

5.3. Wymagania systemowe i sprzętowe

Do przeprowadzenia analizy wydajności pod kątem czasu wykonania testów Cucumber zostały wykorzystane urządzenia o parametrach przedstawionych w tabeli 2.

Tabela 2. Wykorzystane urządzenia do celów badawczych.

Numer urządzenia[1]	Model	Procesor	Pamięć RAM	System operacyjny
NR1	Lenovo IdeaPad Z580	Intel Core i3-2350M	6 GB	Microsoft Windows 7 Professional (x64)
NR2	HP EliteBook 840 G3	Intel Core i5-6300U	16 GB	Microsoft Windows 10 Enterprise (x64)

5.4. Aplikacje testowe

Na potrzeby analizy zostałyprzygotowane trzy identyczne aplikacje w językach Java i Kotlin. Kotlin w porównaniu do Java powstał dość niedawno, w 2011 roku, właśnie głownie z tego powodu taki język został wybrany do badania pod kątem testowania za pomocą narzędzia Cucumber.

Pierwsza aplikacja napisana została w czystym języku programowym bez użycia dodatkowych bibliotek, a jej funkcjonalność polega na zwróceniu sumy kosztów produktu w zależności od podanej liczby produktu. Scenariusz w Cucumber do testowania poprawnego działania danej funkcjonalności przedstawiony jest na przykładzie 1.

Przykład 1. Przypadek testowy №1

```
Feature: Check a product price
Scenario Outline: Check price of milk
Given the price of the "milk" is 2 polish złoty
When the system checks<count> of the "milk"
Then the price should be <bill>polishzłotych
And coś jeszcze, co możemy sprawdzić
Examples:
  | count  | bill |
  |   1    |  2   |
  |   2    |  4   |
  |   5    |  3   |
```

Działanie drugiej aplikacji polega na wyszukiwaniu pewnej informacji w Internecie oraz wyświetlaniu wyniku wyszukiwania w przeglądarce internetowej. Do realizacji wszystkich czynności oprócz samego języka programowaniazostały użyte również biblioteki JUnit oraz Selenium. Logika scenariusza Cucumber dla konkretnego przypadku opisana jest na przykładzie 2.

Przykład 2. Przypadek testowy №2

```
Feature: Google searching
As an user,
I want to search some information in Google,
So that I can find new interesting things.
Scenario: Google searching information
about Lublin University of Technology
Given a chrome browser is on the Google page
When a userenteredphrase "Lublin University of Technology"
Then areshownresults for "Lublin University of Technology"
```

Logika przypadku testowego w trzeciej aplikacjipolega na przetestowaniu formularza do autoryzacji użytkownika w pewnym systemie. Po podaniu przez użytkownika poprawnego loginu i hasła strona z autoryzacją musi przekierować go na stronę główną, a w przeciwnym przypadku na stronę z możliwością odzyskania hasła. W porównaniu do poprzedniej aplikacjido testów zostały dodane również pliki html. Odpowiedni scenariusz dla testów Cucumber został opisany w przykładzie 3.

Przykład 3. Przypadek testowy №3

```
Feature: Login to a web page
@single
Scenario: Successful Login
Given I open chrome browser
When I on the login.html page
And I type my username as Illia and password as 12345
And I click on Sign in button
Then I suppose to see greetings for Illia
@multiple
Scenario Outline:  Successful Login
Given I open chrome browser
When I on the login.html page
And I type my username "<username>"
and password "<password>"
And I click on Sign in button
```

[1] *numer jest wykorzystywany w procesie opisu badania*

```
Then I suppose to see greetings for
"<name>"
Examples:
      | username | password | name  |
      | name1    | pass1    | name1 |
      | name2    | pass2    | name1 |
@reset_multiple
Scenario Outline:  Unsuccessful Login
Given I open chrome browser
When I on the login.html page
And I type my username "<username>"
and password "<password>"
And I try to Sign in
And My password is incorrect and I
forgot my correct password
Then I redirected on the reset.html
Examples:
      | username | password |
      | name1    |   pass1    |
| name2    |   pass2    |
```

5.5. Wyniki badania

Niniejszy rozdział koncentruje się na porównaniu czasu wykonania testów w oparciu o Cucumber, aby wyjaśnić, z którym językiem programowania z rodziny JVM narzędzie może opracowywać testy bardziej wydajnie pod kątem czasu wykonania testu.

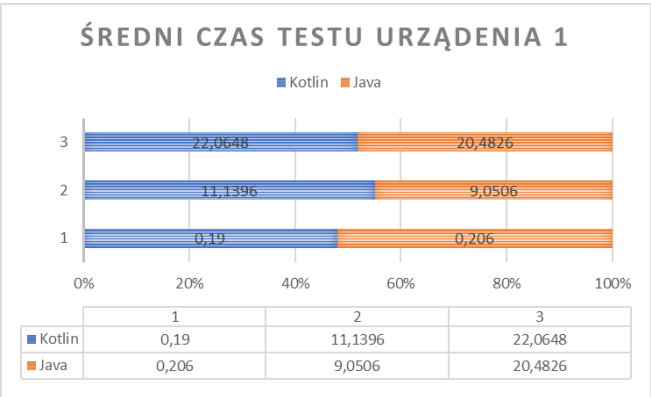
Badanie polegało na uruchomieniu każdego przypadku testowego 5 razy w Java i Kotlin na dwóch urządzeniach NR1 i NR2, wyniki testów są przedstawione w sekundach na tabelach3 i 4.

Tabela 3. Dane otrzymane na urządzeniu NR1.

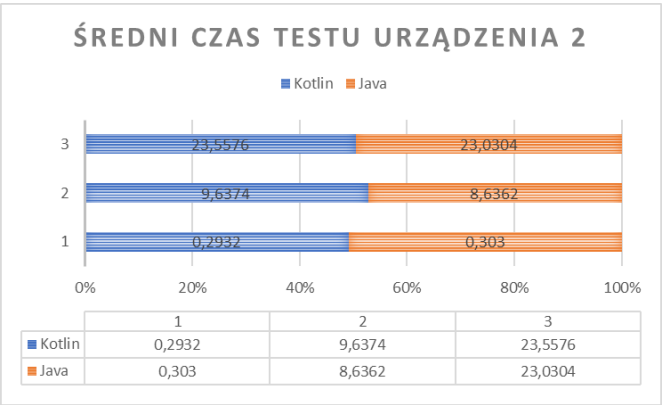
Przypadek testowy	**Czas wykonania testu w Kotlin w sekundach**	**Czas wykonania testu w Java w sekundach**
№1	0,182	0,215
	0,193	0,198
	0,194	0,219
	0,169	0,182
	0,212	0,216
	Średni czas wykonania testu 0,19	Średni czas wykonania testu 0,206
№2 (JUnit, Selenium)	11,274	9,211
	12,602	9,272
	11,656	8,962
	9,795	8,932
	10,371	8,876
	Średni czas wykonania testu 11,1396	Średni czas wykonania testu 9,0506
№3 (JUnit, Selenium, obsługa plików HTML)	21,254	21,88
	22,338	20,704
	23,215	19,924
	21,405	19,802
	22,112	20,103
	Średni czas wykonania testu 22,0648	Średni czas wykonania testu 20,4826

Tabela4. Dane otrzymane na urządzeniu NR2.

Przypadek testowy	**Czas wykonania testu w Kotlin w sekundach**	**Czas wykonania testu w Java w sekundach**
№1	0,282	0,337
	0,316	0,323
	0,277	0,287
	0,288	0,262
	0,303	0,306
	Średni czas wykonania testu 0,2932	Średni czas wykonania testu 0,303
№2 (JUnit, Selenium)	11,411	8,227
	8,77	8,359
	10,062	8,28
	8,813	10,11
	9,131	8,205
	Średni czas wykonania testu 9,6374	Średni czas wykonania testu 8,6362
№3 (JUnit, Selenium, obsługa plików HTML)	23,044	23,981
	24,651	22,666
	24,323	22,902
	22,957	23,48
	22,813	22,123
	Średni czas wykonania testu 23,5576	Średni czas wykonania testu 23,0304



Rys. 1. Średni czas wykonania testów na NR1.



Rys. 2. Średni czas wykonania testów na NR2.

Pomimo różnego czasu wykonania konkretnego testu na urządzeniach, wynik pod kątem zwycięstwa jest dokładnie ten

sam. W przypadku testowym №1 na obu urządzeniach Kotlin prędzej wykonuje swoje zadanie. Ciekawym faktem jest to, że urządzenie o znacznie słabszych parametrach, ale z systemem operacyjnym Windows 7 szybciej opracowało pierwszy oraz trzeci przypadek testowy. Ciekawie czy to jest kwestia systemu operacyjnego?! W kolejnych przypadkach testowych №2 oraz №3 Java się okazała szybszą od Kotlin.

6. Wnioski

Podsumowując wyniki eksperymentuz rozdziału 6 można przypuszczać, że Kotlin prędzej opracowuje testy Cucumber w aplikacjach, gdy nie potrzebują one dodatkowych bibliotek do swojego działania, odpowiednio Java jest bardziej zoptymalizowanym narzędziem, jeżeli chodzi o współpracę z innymi bibliotekami, więc kwestia wyboru konkretnego języku do napisania wydajniejszych testów w Cucumber pod kątem czasu wykonania testów to kwestia konkretnego przypadku testowego.

Nawiązując dohipotezy badawczejnależy podsumować, iż jednak istnieje różnica w czasie wykonania testów w obu językach, nawet zwracającuwagę na faktdziałania w obrębietej samej maszyny wirtualnej Java, a więc hipoteza badawcza nie została potwierdzona.

W artykule przedstawiono analizę zastosowania narzędzia Cucumber w testowaniu aplikacji.Dużym plusem narzędzia jest zrozumiały język testów nawet dla osób nietechnicznych, łatwa integracja z biblioteką JUnit, Selenium, a także obsługa wyrażeń regularnych oraz możliwość uruchamiania jednego testu z różnymi zestawami danych.

Literatura

[1] Chelimsky, David, et al. The RSpec Book: Behaviour-Driven Development with

[2] Full-scale Software Engineering / Current Trends in Release Engineering, Faculty of Mathematics, Computer Science, and Natural Sciences, Research Group Software Construction, 2016.

[3] Semi-automatic Test Case Generation, Olav Undheim, Norwegian University of Science and Technology Department of Computer and Information Science, 2011.

[4] Dokumentacja Cucumber, https://cucumber.io/ [21.10.2018]

[5] Ian Dees, Matt Wynne, Aslak Hellesoy, Cucumber Recipes, 2013.

[6] Seb Rose, Matt Wynne, Aslak Hellesøy, The Cucumber for Java Book, 2015.

[7] Matt Wynne, Aslak Hellesøy, Steve Tooke, The Cucumber Book, Second Edition, 2017.

[8] Gáspár Nagy, Seb Rose, The BDD Books – Discovery, 2017.

[9] Daniel J. Mosley, Bruce A. PoseyJust, Enough Software Test Automation, 2002.

[10] Dr. Winston W. Rovce , „Managing the Development of Large Software Systems”, 1970

[11] Paweł Marek, „Weryfikacja i automatyzacja procesu testowania oprogramowania”, CORE Magazine, 2010.